Spring Boot Application in Docker

Spring Boot Overview

- **What is Spring Boot?**
  - Framework for building stand-alone Java applications.
  - Simplifies development with embedded servers and auto-configuration.
- **Why use Spring Boot with Docker?**
  - Consistency across environments.
  - Easy deployment and scaling.
  - Portable, lightweight, and efficient.

Prerequisites

- **What you need**:
  1. **Java 17 SDK**: Installed and configured.
  2. **Spring Boot CLI**: Optional but useful for scaffolding projects.
  3. **Docker**: Installed and running on your machine.

4. **Maven**: For build automation.

Creating a Simple Spring Boot Application

- **Step 1**: Initialise a Spring Boot project with Maven.
    - Use Spring Initializr to generate the project structure.
    - Choose **Maven** as the project, Java version, and **Spring Web** dependency.
    - **Group**: `com.example`, **Artifact**: `song-suggester`
    - **Download** the project and unzip it.

Writing the Random Song Suggester App

- **Step 2**: Create the Song Suggester logic.
    - In `src/main/java/com/example/songsuggester/SongSuggesterController.java`:

```
package com.example.songsuggester;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestControll
er;
import org.springframework.web.client.RestTemplate;
import org.json.JSONObject;

import java.util.Random;

@RestController
public class SongSuggesterController {

    @GetMapping("/suggest")
    public String suggestSong() {
        String apiUrl = "https://itunes.apple.com/search?te
rm=pop&limit=10";
        RestTemplate restTemplate = new RestTemplate();
        String result = restTemplate.getForObject(apiUrl, S
tring.class);
```

```
        // Parse the JSON response
        JSONObject jsonObject = new JSONObject(result);
        var tracks = jsonObject.getJSONArray("results");

        // Randomize the selection
        Random rand = new Random();
        int randomIndex = rand.nextInt(tracks.length());
        var randomTrack = tracks.getJSONObject(randomInde
x);

        // Extract the song and artist name
        String song = randomTrack.getString("trackName");
        String artist = randomTrack.getString("artistNam
e");

        return "Today's song suggestion: " + song + " by "
+ artist;
    }
}
```

Running the Application Locally

- **Step 3**: Build and run the Spring Boot application.
  - Navigate to the project folder.
  - Run the Maven build command:

    ```
    mvn clean install
    ```

  - Start the Spring Boot application:

    ```
    mvn spring-boot:run
    ```

  - **Access the application**:
    - Open your browser and go to: `http://localhost:8080/suggest`

Preparing the Application for Docker

- **Step 4**: Write a `Dockerfile`.
    - In the root of your project directory, create a `Dockerfile`:

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17-jdk-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the project JAR file into the container
COPY target/song-suggester-0.0.1-SNAPSHOT.jar app.jar

# Expose the port the app runs on
EXPOSE 8080

# Run the JAR file
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Building the Docker Image

- **Step 5**: Build the Docker image from the Dockerfile.
    - Run this command in the project directory where the Dockerfile is located:

        ```
        docker build -t song-suggester .
        ```

    - This command builds the image with the name `song-suggester` using the current directory (`.`).

Running the Docker Container

- **Step 6**: Run the Docker container.
    - Run this command to start the container and map port 8080:

        ```
        docker run -p 8080:8080 song-suggester
        ```

- **Test the application**: Open your browser and go to `http://localhost:8080/suggest` to see the random song suggestion.

Docker Best Practices

- **Best Practices for Docker**:
  1. **Use minimal base images**: For smaller, faster containers.
  2. **Use multi-stage builds**: To reduce the final image size.
  3. **Include a `.dockerignore`**: To exclude unnecessary files during the build.

Useful Links and Resources

What is a Container? | Docker

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to

https://www.docker.com/resources/what-container

Play with Docker

Play with Docker (PWD) is a project hacked by Marcos Liljedhal and Jonathan Leibiusky and sponsored by Docker Inc.

https://labs.play-with-docker.com/

Get started

Get started with Docker

https://docs.docker.com/get-started/

Educational resources

Get started resources learning docker

https://docs.docker.com/get-started/resources/